On Improving the Performance of an Iterative Linear Solver

Allison Baker
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Joint work with E.R. Jessup, J.M. Dennis
Department of Computer Science
University of Colorado

Portions of this work were performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Can we solve large, sparse systems of linear equations more efficiently?

- linear system: $\mathbf{Ax} = \mathbf{b}$

  ($\mathbf{A}$ is large, sparse, and nonsymmetric)

- costs: (1) floating-point operations (FLOPs)

  (2) movement of data through memory

- iterative linear solver: GMRES(m)

  (restarted Generalized Minimum Residual method)

Motivation

- linear systems are often time-consuming to solve
- GMRES(m) convergence can be slow
- moving data is expensive
  - iterative linear solvers:

    access $\mathbf{A}$ once per iteration loop for $\mathbf{A} * \mathbf{x}$

Memory costs impact performance

memory hierarchy component	typical CPU latency value
L1 cache	1 - 3 cycles
L2 cache	6 - 20 cycles
main memory	60 - 140 cycles
disk	10,000 cycles!!!

- getting worse: improvement in

  microprocessor performance vs. memory performance:
  $\sim$ 60% per year $<$ 10% per year

- memory costs >> FLOP costs for large problems

Approach for improving GMRES(m)

(1): FLOPs

- improve convergence behavior $\Rightarrow$ reduce the number of iterations

(2): data movement

- reformulate algorithm to reduce data movement

$$\begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{b} \\ \\ \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{B} \\ \\ \end{bmatrix}$$

- "smart" implementation

Accelerating convergence

- augmented methods

  (Morgan '95, '00, Chapman and Saad '97)

  – add additional vectors to Krylov subspace

- nested Krylov (truncated)

  (Van der Vorst and Vuik '94, de Sturler '96 and '99)

  – approximately solve residual equation ($Ae = r_i$)

- LGMRES

  (Baker, Jessup, Manteuffel)

  – combination of the approaches above

What should we choose for B for GMRES(m)? ($\mathbf{Ax = b \Rightarrow AX = B}$)

- goal:
  balance memory-usage and favorable numerical properties

- observation:
  restarted GMRES(m) throws away useful information with every restart, which slows convergence

- our approach:
  compensate for this loss $\rightarrow$ put the lost information in B!

$\mathbf{Ax = b \Rightarrow AX = B}$

GMRES(m):

- restart cycle $\mathbf{i}$: $\mathbf{x_i = x_{i-1} + c}, \quad \mathbf{c \in \mathcal{K}_m(A, r_{i-1})}$
- ideally: $\mathbf{c = x_{exact} - x_{i-1} \longrightarrow x_i = x_{exact}}$
- error approximation: $\mathbf{z_i \equiv x_i - x_{i-1}}$

idea:

- put the error approximation vector in $\mathbf{B}$: $\mathbf{B = [\, b\,,\, z_i\,]}$

A new algorithm: B-LGMRES

B-LGMRES(m, 1):

- restart cycle $\mathbf{i+1}$: $\mathbf{Ax = b \Rightarrow AX = B}$

  $\mathbf{X = [\, x_i\,,\ 0\,]}$ (initial guess)

  $\mathbf{B = [\, b\,,\ z_i\,]}$ (right-hand side)

  i.e., augment $\mathbf{b}$ with an error approximation

  $(\mathbf{z_i \equiv x_i - x_{i-1}})$

- $\underline{\mathbf{x_{i+1} \in x_i + \mathcal{K}_m(A, r_i)}} + \mathcal{K}_\mathbf{m}(\mathbf{A}, \mathbf{z_i})$

  $\|$

  GMRES(m)

B-LGMRES: an efficient implementation

Matrix-multivector multiply (Kaushik '99):

multiply $\mathbf{A}$*4 vectors in $\approx$ 1.5 the time for $\mathbf{A}$*1 vector

- process the vectors in groups or "multivectors"
- interleave the vector components in memory

  e.g., $\mathbf{v_1(j)}$, $\mathbf{v_2(j)}$, $\mathbf{v_3(j)}$, $\mathbf{v_4(j)}$ are adjacent (row-wise)
- unroll the inner matrix multiplication loop
  (across multivector elements)

Evaluating the new method:

B-LGMRES(15, 1) vs. GMRES(30):

- equal approximation space size (30)
- variety of problems from several test collections
- problem size: $n = 7,740$ to $1,709,982$
  ($nnz = 79,566$ to $1,717,792$)
- either ILU preconditioner or no preconditioner
- Argonne's PETSc with modifications

Some results...

$\Rightarrow$ the new method (B-LGMRES) is generally faster

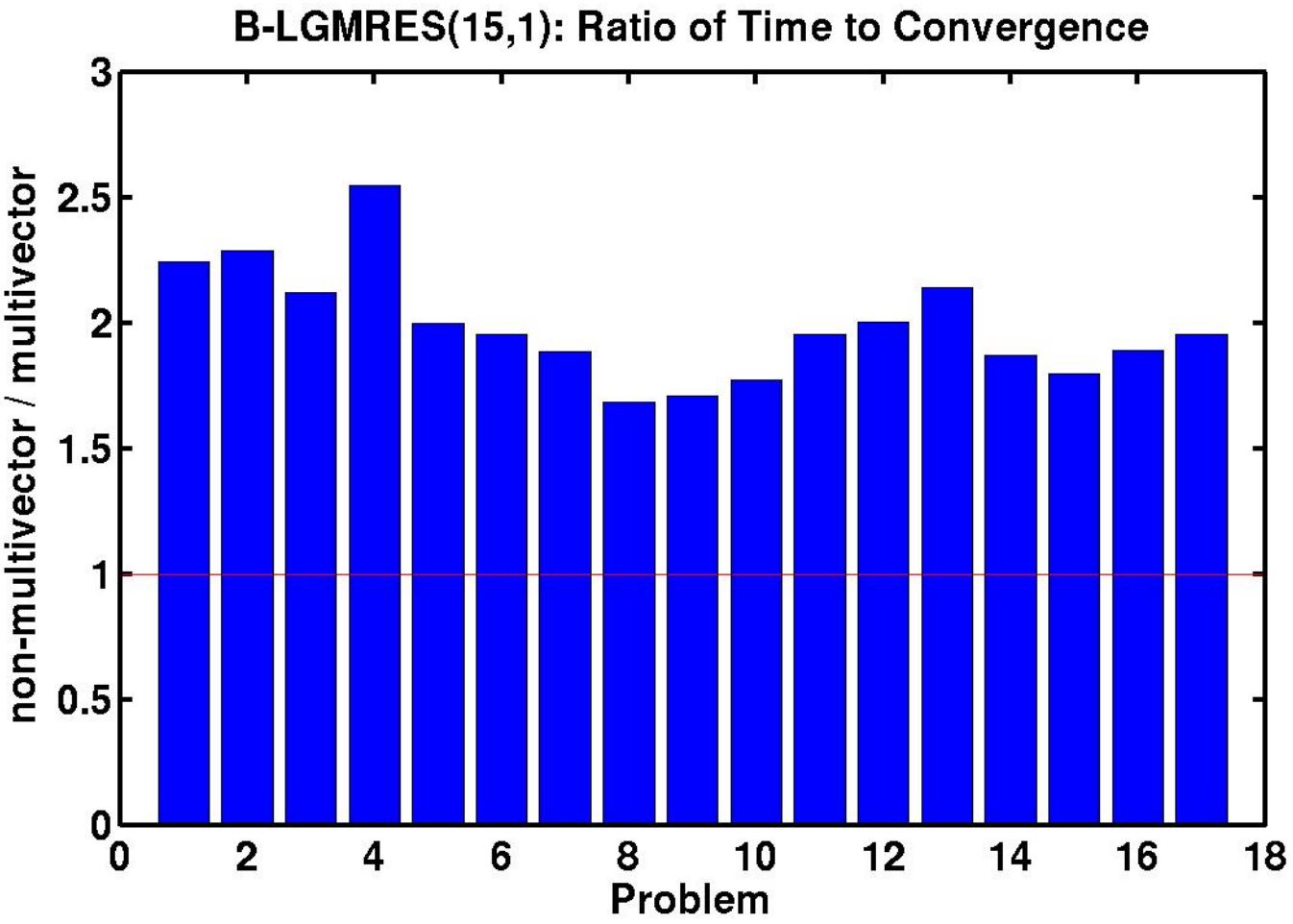
Problem 12: 2-D fluid flow

size: n=13,535, nnz=390,607

method	execution time	accesses of $\mathbf{A}$	matrix-vector multiplies
GMRES(30)	$165.9 \pm .6$ s	1960	1960
B-LGMRES(15,1)	$78.5 \pm .2$ s	1004	2008

$\Rightarrow$ execution time correlates to number of accesses of $\mathbf{A}$
(related to memory usage, not floating-point operations)

Importance of an efficient implementation



Conclusions

(1): FLOPs

- accelerate GMRES(m) convergence by preventing repetitiveness

(2): data movement

- algorithmic changes: changing b to B
- implementation: basic programming tricks

$\Rightarrow$ modifying a linear solver to reduce data movement is a viable approach to gaining efficiency!